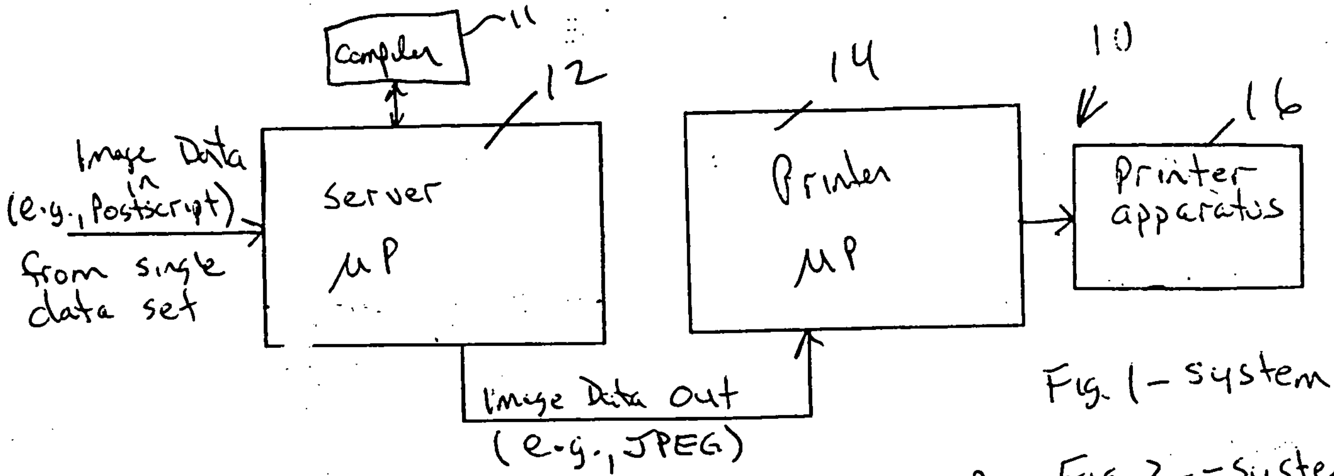
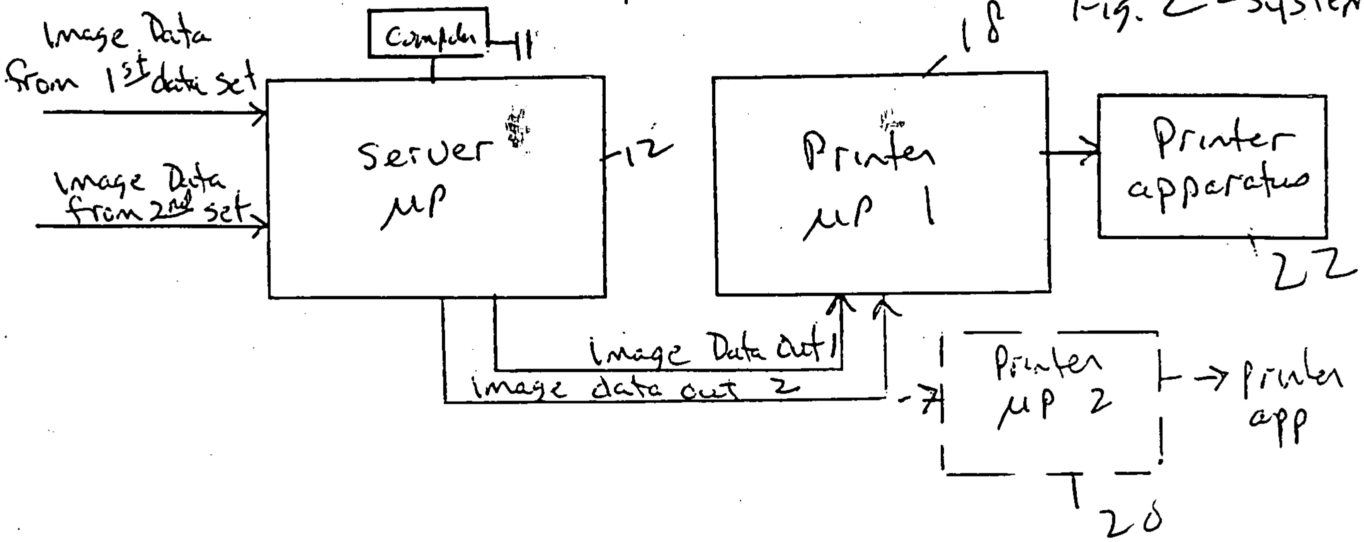

Compiler
11
12
14
10
16
Image Data in (e.g., Postscript) from single data set
Server µP
Printer µP
printer apparatus
Image Data Out (e.g., JPEG)

Fig. 1 – system

Fig. 2 – system



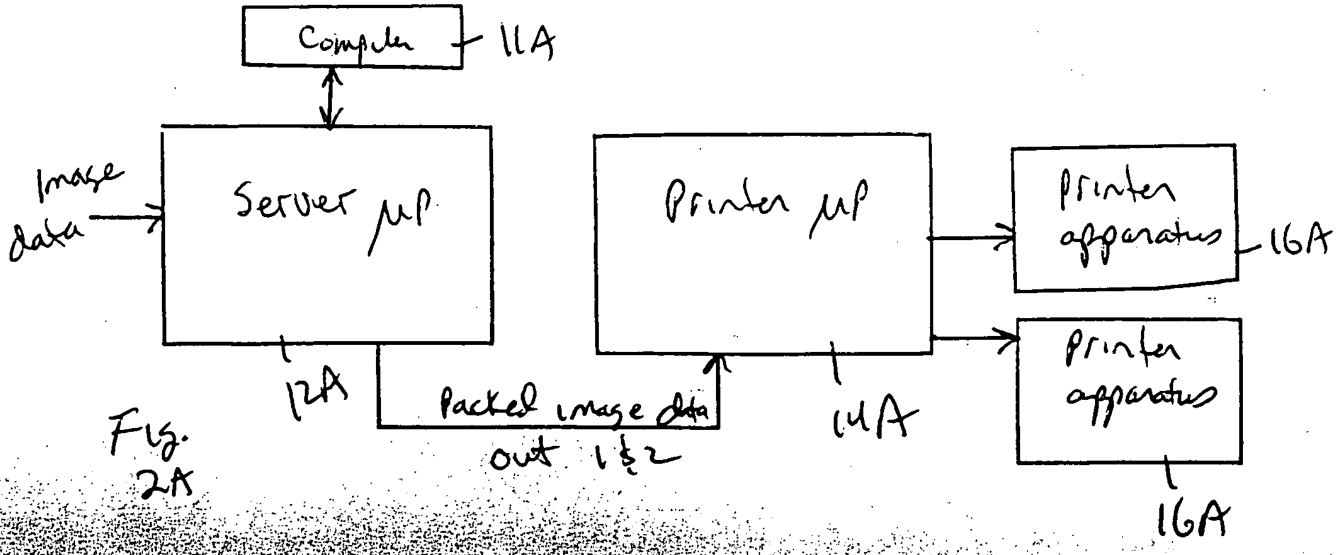


Fig. 2A

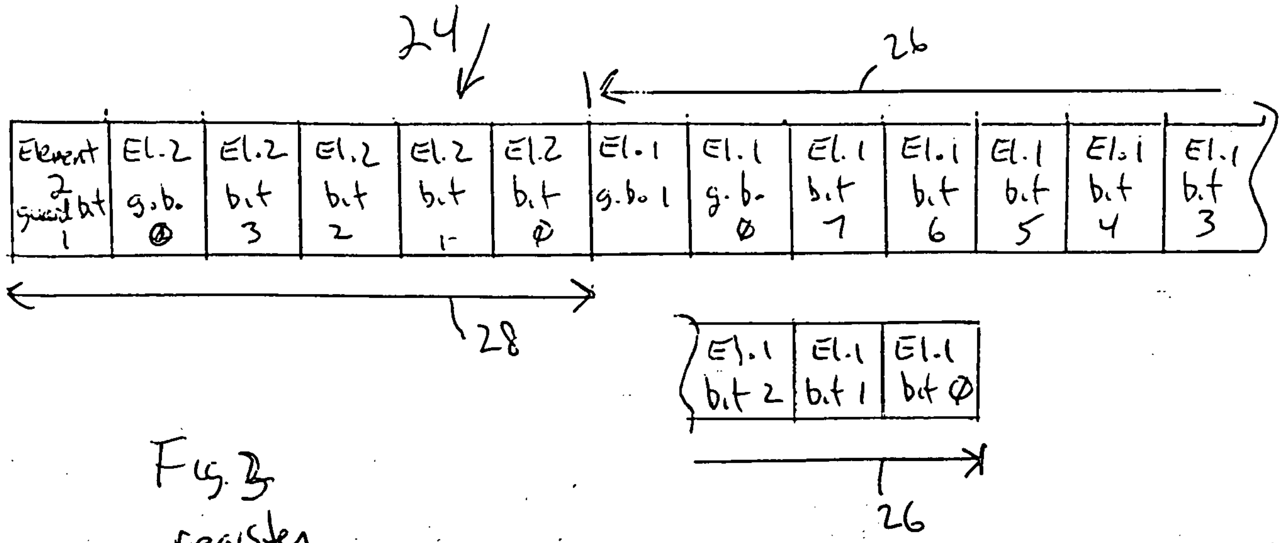


Fig. 3
register

Fig. 4 overall flow

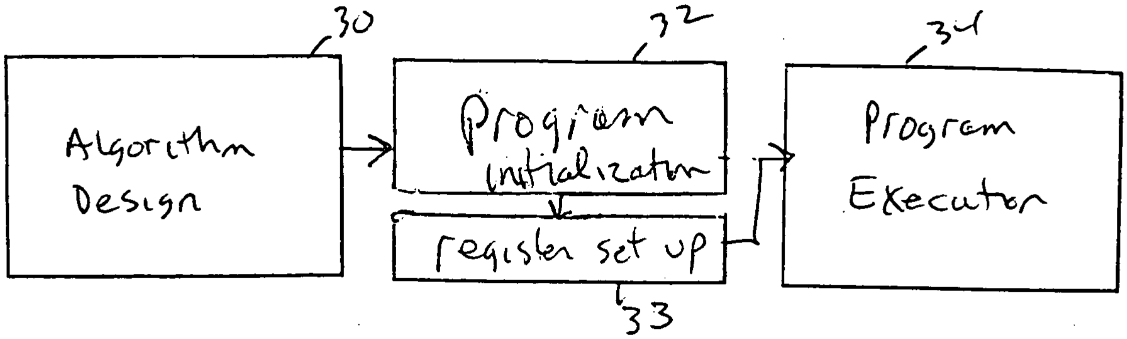
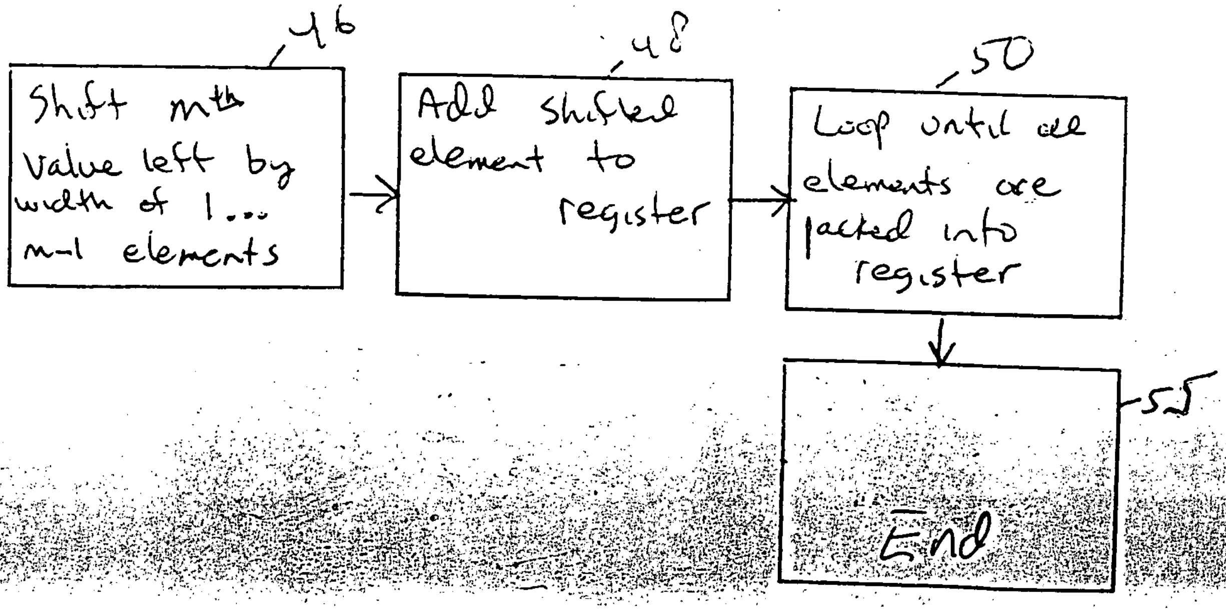


Fig. 6 register set up


46
Shift mth value left by width of 1... m-1 elements
48
Add shifted element to register
50
Loop until all elements are packed into register
52
End

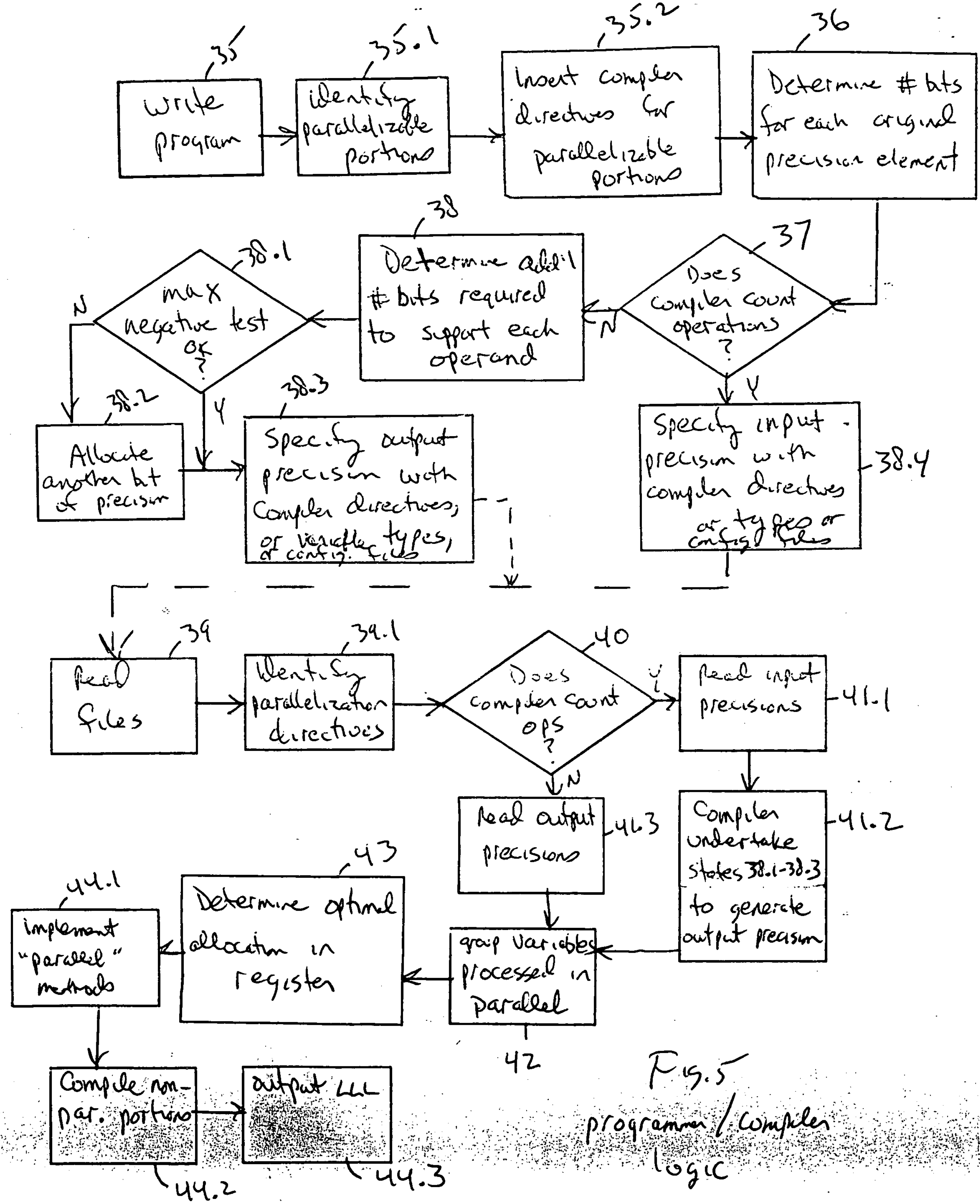

35
Write program
35.1
identify parallelizable portions
35.2
Insert compiler directives for parallelizable portions
36
Determine # bits for each original precision element
37
Does compiler count operations ?
N
Y
38
Determine add'l # bits required to support each operand
38.1
max negative test ok ?
N
Y
38.2
Allocate another bit of precision
38.3
Specify output precision with compiler directives, or variable types, or config. files
38.4
Specify input precision with compiler directives or types or config files
39
Read files
39.1
Identify parallelization directives
40
Does compiler count ops ?
Y
N
41.1
Read input precisions
41.2
Compiler undertake states 38.1-38.3 to generate output precision
41.3
Read output precisions
42
group variables processed in parallel
43
Determine optimal allocation in register
44.1
implement "parallel" methods
44.2
Compile non-par. portions
44.3
output LLL

Fig. 5
programmer/compiler logic

Fig. 5A
Initialization

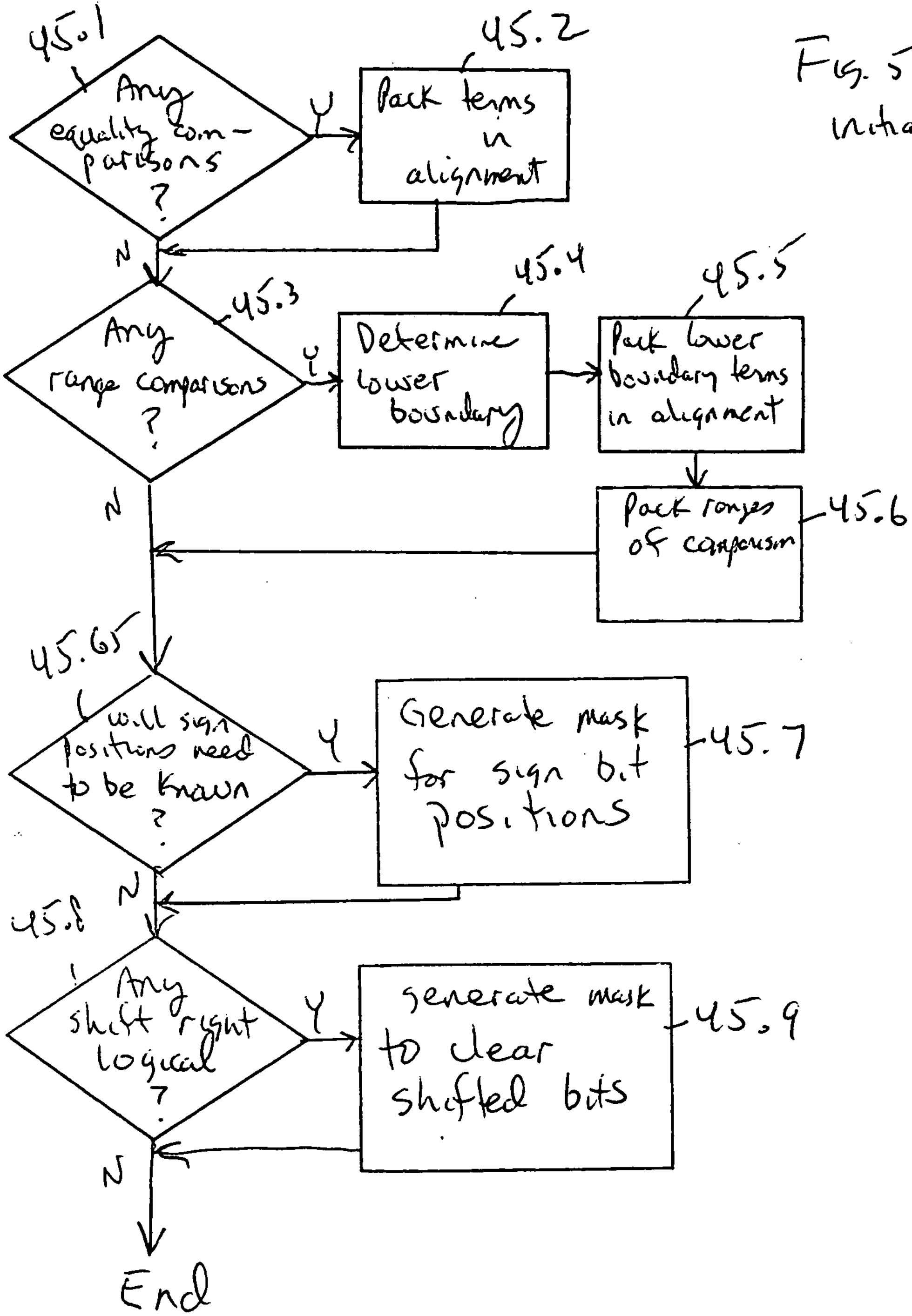
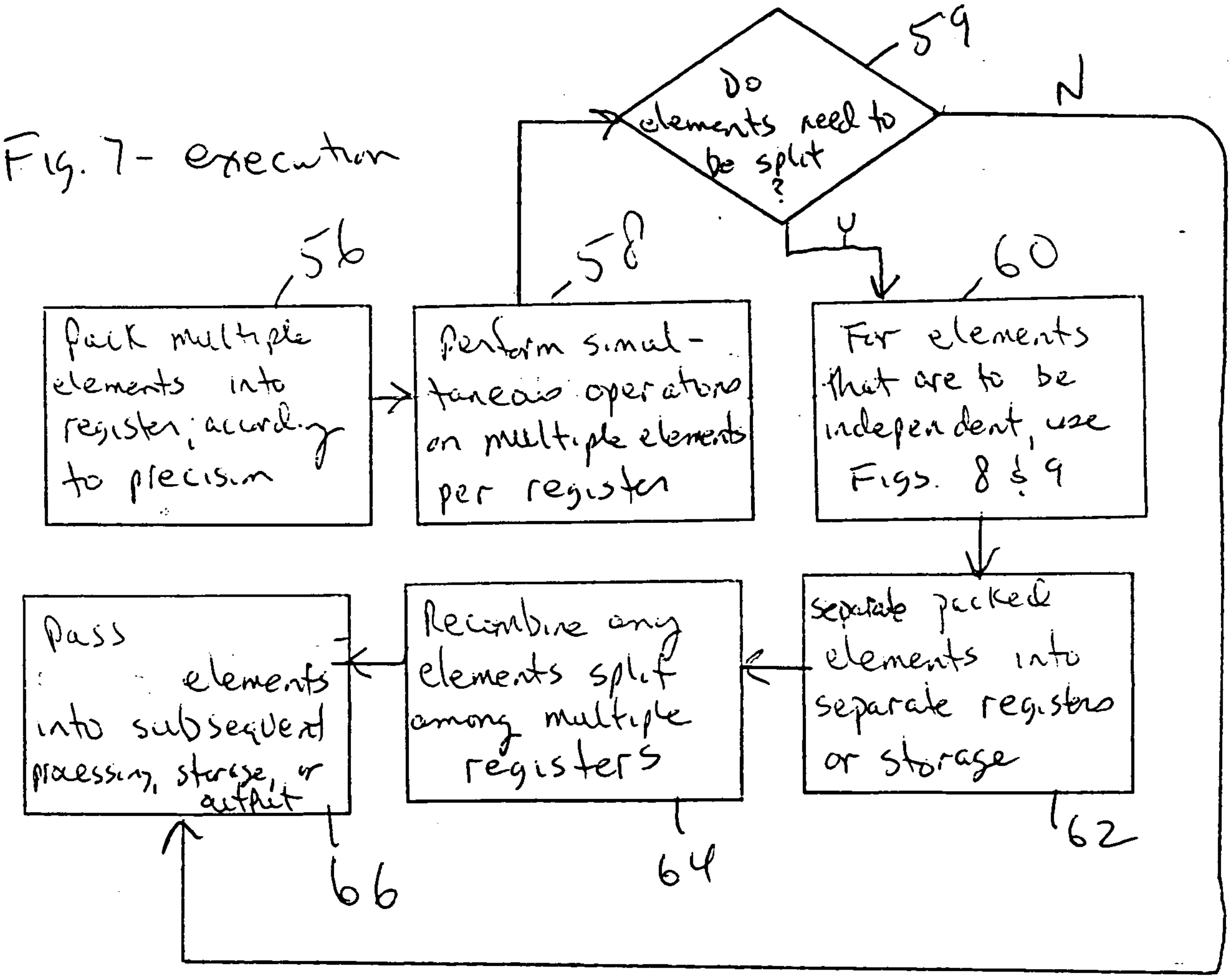

45.1
Any equality comparisons ?
Y
45.2
Pack terms in alignment
N
45.3
Any range comparisons ?
Y
45.4
Determine lower boundary
45.5
Pack lower boundary terms in alignment
45.6
Pack ranges of comparison
N
45.65
will sign positions need to be known ?
Y
45.7
Generate mask for sign bit positions
N
45.8
Any shift right logical ?
Y
45.9
generate mask to clear shifted bits
N
End

Fig. 7- execution


56
Pack multiple elements into register, according to precision
58
Perform simultaneous operations on multiple elements per register
59
Do elements need to be split ?
N
Y
60
For elements that are to be independent, use Figs. 8 & 9
62
separate packed elements into separate registers or storage
64
Recombine any elements split among multiple registers
66
Pass elements into subsequent processing, storage, or output

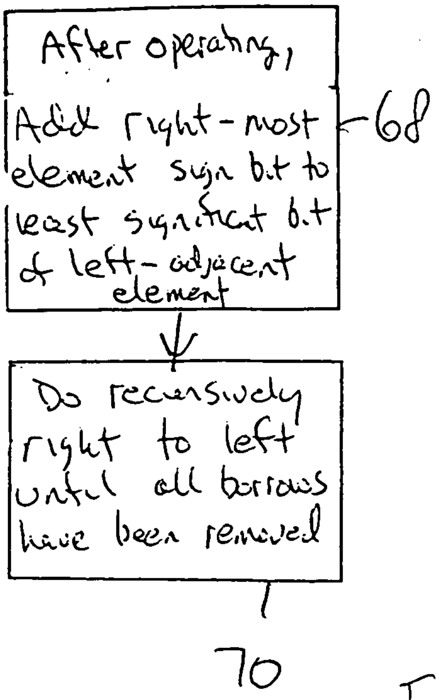

After operating,
Add right-most element sign bit to least significant bit of left-adjacent element
68
Do recursively right to left until all borrows have been removed
70

Fig 8

logic for independent elements

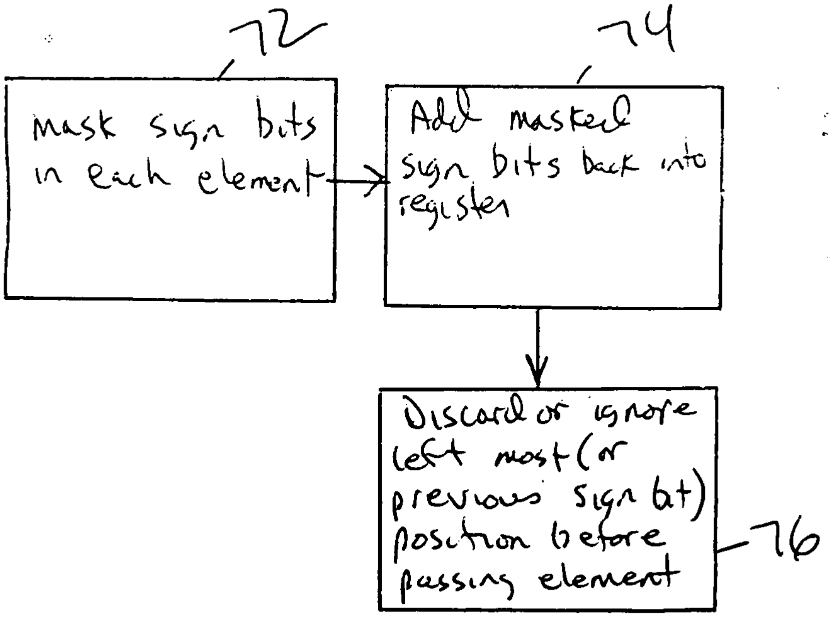

72
74
mask sign bits in each element
Add masked sign bits back into register
Discard or ignore left most (or previous sign bit) position before passing element
76

Fig. 9- alto logic for independent elements

Fig. 10-logic

for determining whether element contained around zero

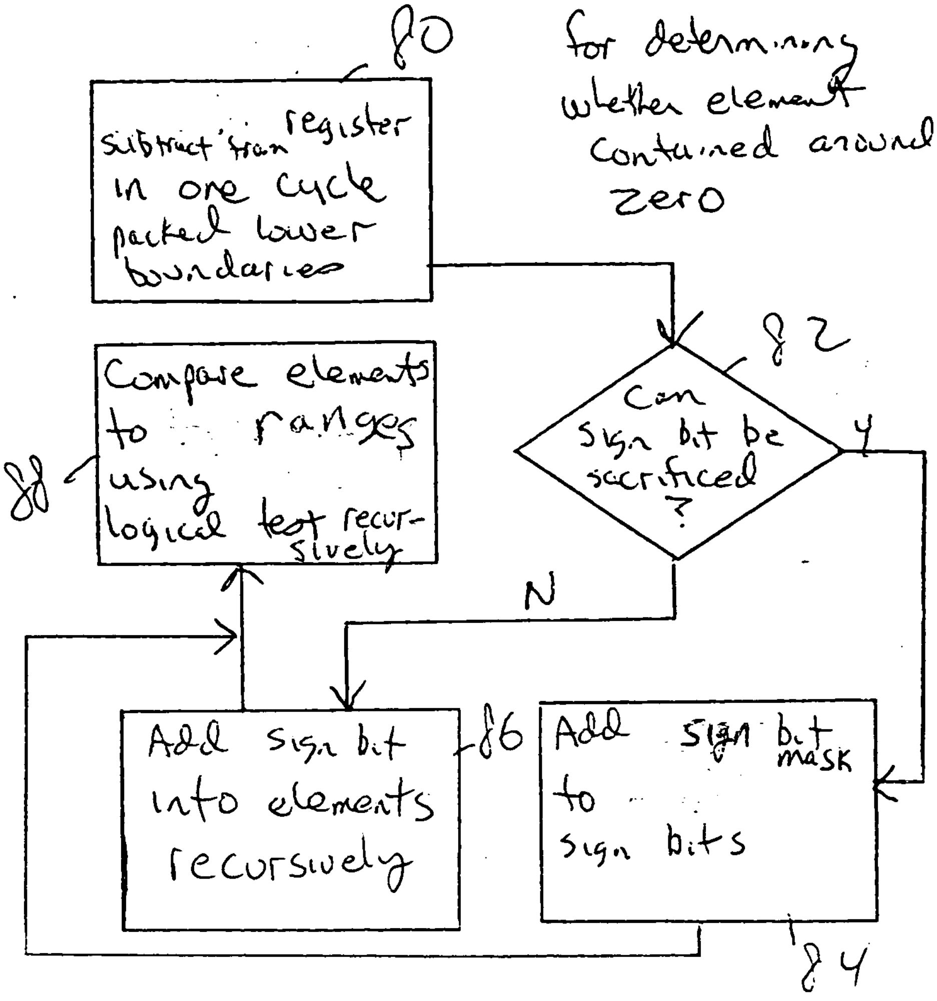

80
subtract from register in one cycle packed lower boundaries
82
Can sign bit be sacrificed ?
Y
N
84
Add sign bit mask to sign bits
86
Add sign bit into elements recursively
88
Compare elements to ranges using logical test recursively

Fig. 12 - logic for arithmetic compares:

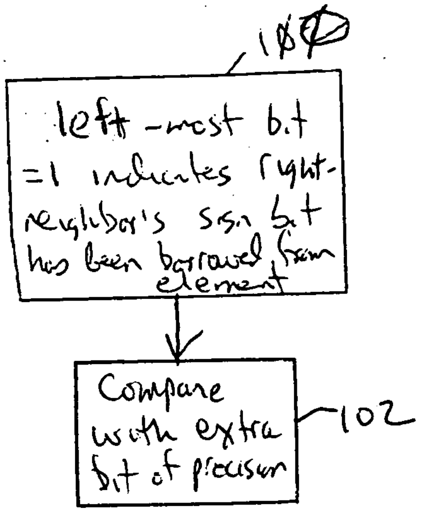
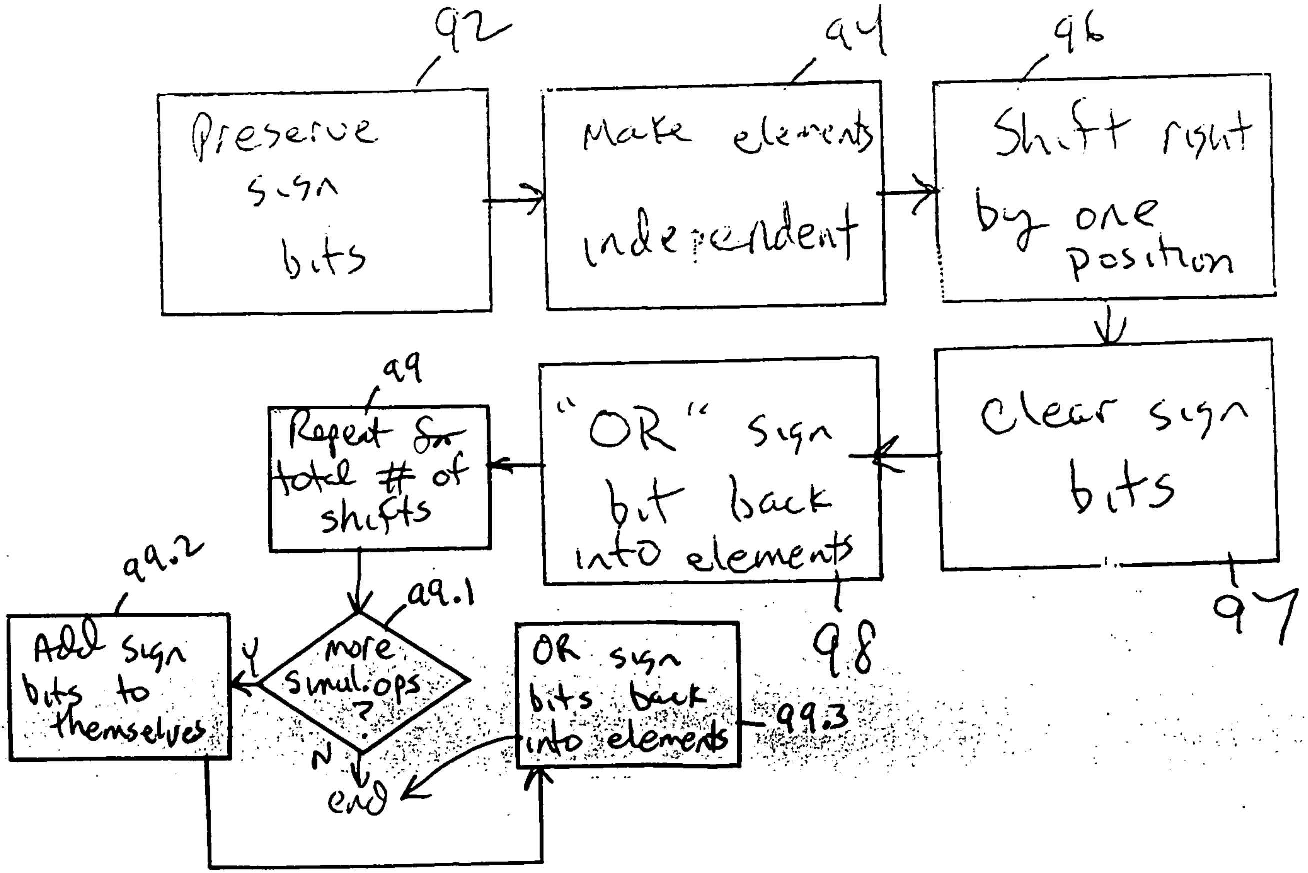

100
left-most bit =1 indicates right-neighbor's sign bit has been borrowed from element
Compare with extra bit of precision
102

Fig. 11


92
Preserve sign bits
94
Make elements independent
96
Shift right by one position
Clear sign bits
97
"OR" sign bit back into elements
98
99
Repeat ~~for~~ total # of shifts
99.1
more simul.ops ?
Y
N
end
99.2
Add sign bits to themselves
OR sign bits back into elements
99.3